CCC 2020
Proceedings of the Creative Construction e-Conference (2020) 026
Edited by: Miroslaw J. Skibniewski & Miklos Hajdu
https://doi.org/10.3311/CCC2020-026

Formal Modeling of Smart Contracts for Quality Acceptance in Construction

Da Sheng[1], Hanbin Luo[2] and Botao Zhong[3]

[1] *Huazhong University of Science & Technology, Wuhan, China, shengda@hust.edu.cn*
[2] *Huazhong University of Science & Technology, Wuhan, China, luohbcem@hust.edu.cn*
[3] *Huazhong University of Science & Technology, Wuhan, China, dadizhong@hust.edu.cn*

Abstract

The coding and deployment of smart contracts in the construction industry are challenging because of the gap between the generality of existing modeling approaches for such contracts and the pertinence of business logic to construction management. This research proposes a formal model for smart contracts in the context of quality acceptance in the construction industry to reduce the threshold for applying smart contract technology. First, a conceptual scenario of smart contract-based quality acceptance in construction is analyzed. Second, a finite state machine-based model is proposed to formalize smart contracts for quality acceptance. Lastly, a Hyperledger-based case study is performed to demonstrate the performance of the proposed formal model. This study contributes to the industrial application of formal modeling approaches for smart contracts in the field of construction.

Keywords: blockchain, finite state machine, quality acceptance, quality management, smart contract

1. Introduction

Blockchains are one of the most high-profile technologies because of its traceability, decentralization, and tamper-proofing capability. The support for smart contract programming enables blockchain systems to automate several business processes in addition to issuing currencies [1]. Various industries devote continuous effort to technological explorations, resulting in the emergence of a batch of preliminary applications. However, the construction industry often responds slowly to innovative technologies, and studies on blockchain and smart contracts are still a novelty in this field. Perera et al. [2] stated that smart contract-enabled blockchain systems can resolve the fragmentation issues caused by the complexity of construction projects and keep the information updated and available for stakeholders in the management process. Li et al. [3] believe that smart contracts have the potential to change the organization operation and reduce disputes. Wang et al. [4] established a Hyperledger-based framework to address the challenges (e.g., poor traceability/timeliness of information) faced by the current precast supply chain and suggested that smart contracts are the sole channel for management personnel to interact with distributed ledgers. Furthermore, blockchain and smart contracts have been demonstrated to have advantages in integrated project delivery [5] and payment security [6].

Based on the functions of blockchain and smart contracts, we posit that their assembly also has advantages in quality acceptance in construction. First, quality compliance inspection, which is crucial for quality acceptance, is a knowledge-intensive task that involves a succession of regulations and industry provisions; this succession renders manual task execution cumbersome and error prone. Smart contracts can semi-automate several checking tasks and reduce the overall time consumption. Second, the substantial quality information generated during construction must be shared and stored carefully. However, a centralized database cannot guarantee data security, and data holders can tamper with the data. A blockchain-based distributed solution that allows several or all participants to have a duplicate of the ledger can effectively avoid such misdeeds. Lastly, quality compliance inspection usually becomes an afterthought in practice due to the lack of constraints on the individuals' responsibility. A system based on blockchain and smart contracts enables all participants to formulate jointly a code of conduct before a project starts and enforce it during construction. A predefined punitive measure is imposed when an inspector fails to complete the agreed operation within the time constraints.

The programming and deployment of smart contracts require specialization; thus, developers must possess computer literacy and professional proficiency. However, such requirement is impractical. Consequently, the advantages of smart contracts are hardly maximized. Formalization is an important topic in smart contract research. Smart contracts are vulnerable to malicious attacks (e.g., the attack on the Decentralized Autonomous Organization in 2016) due to the bugs caused by the negligence or limited knowledge of developers. Formalization approaches can help developers reduce faults in smart contracts and assist in software testing. Singh et al. [7] reviewed several prevalent approaches, including theorem proving, symbolic execution, model checking, formal modeling, and finite state machine (FSM). These approaches were proposed to verify contractual functionalities [8] and alleviate the security [9] and privacy [10] issues of smart contracts. However, these approaches are difficult for quality inspectors to apply because of the gap between the generality of these approaches and the pertinence of quality acceptance in construction. Thus, this study proposes a formal modeling approach for smart contracts in the context of quality acceptance in the construction industry to relax the barrier to smart contract adoption and reduce the perplexity of applying this technology to construction projects.

2. Smart contract-based quality acceptance in construction

A massive amount of quality information is generated from the measuring and testing processes during construction. However, the absence of a uniform system among the stakeholders of a construction project results in substantial "noise" (e.g., blank and tampering) in the collected quality information. Quality management systems that enable the precise documentation and tracing of quality information are desired. Blockchain technology can free quality information from exclusive control and thus promotes trust among project participants. Figure 1 presents the conceptual framework of construction quality blockchain. This blockchain, which is a distributed database for on-chain projects, preserves the complete latest records and documents regarding quality. On-chain information can be continuously utilized for project management and can be accessed by users through application clients.

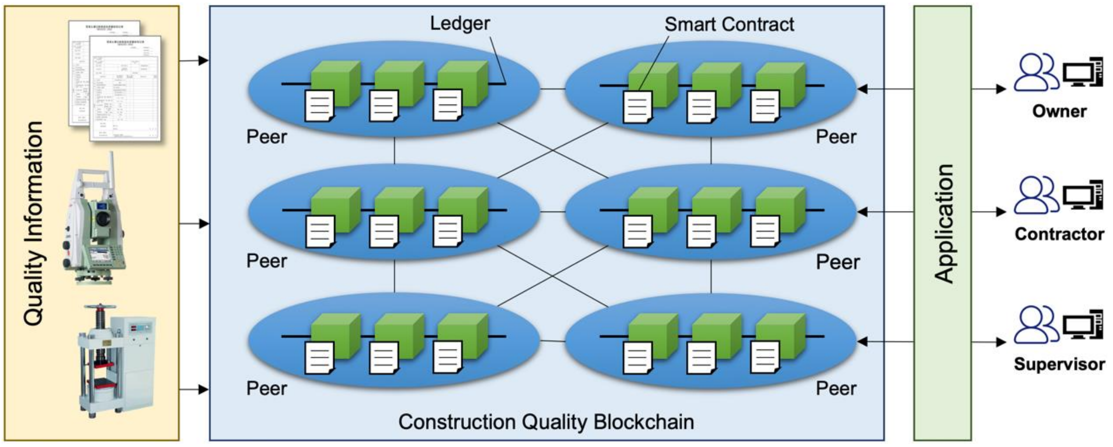


Figure 1. Conceptual framework of construction quality blockchain

Construction quality blockchain network is a peer-to-peer network composed of the stakeholders of construction projects. In such network, each participant serves as a peer. After obtaining quality information, the responsible person is obliged to upload it to the blockchain network. Through a consensus process, the uploaded information is sorted, written in the latest block, and broadcasted. Each peer downloads the block from the network and refreshes its own duplicate ledger to maintain the consistency of quality information across the network. Once updated in the ledger, information cannot be controlled by any person or entity (i.e., cannot be modified and deleted). Users can only implement quality management through the application client, which is the bridge between users and on-chain information. For example, project stakeholders, such as owners, contractors, and supervisors, can create a smart contract-based agreement regarding the responsibilities and constraints for quality management through application interfaces before the project commences. After the consensus process, this smart contract is backed up to each peer (i.e., the agreement is endorsed by the entire network). Subsequently, the participants fulfil the agreements for project delivery. For example, the contractor performs construction tasks in accordance with the schedule and uploads quality information. Meanwhile, the owner and supervisor conduct timely compliance checking and acceptance.

Smart contracts are not a new concept and have been existing for more than 25 years. In 1994, Szabo [11] defined a smart contract as "a computerized transaction protocol that executes the terms of a contract" and aims to minimize exceptions and intermediaries. However, the technological development of smart contracts was postponed because of the difficulty of establishing a trusted cooperation environment in a centralized management system. The emergence of blockchains provides a low-budget opportunity for developing and applying smart contracts. A smart contract contains the trigger conditions for responses and required actions. Once deployed, it can continuously monitor data fluctuation in the chain or external source. Moreover, it can automatically execute the predefined actions when triggered. In conventional business systems, multiple manual executions and intermediate interventions are inevitable for the same functions. Smart contracts can handle complex application scenarios because they are supported by Turing complete programming languages, such as Go for Hyperledger and Solidity for Ethereum. However, in several industries, such as construction, the large-scale application of smart contracts is challenging because practitioners must adapt to brand-new, domain-specific programming languages and logic.

The assembled technology of blockchain and smart contracts has the potential to semi-automate the procedure of quality acceptance which is a concurrent activity of the construction process, and preserve quality information in a tamper-proof and traceable manner. Figure 2 displays a graphical description of the conceptual scenario for smart contract-based quality acceptance in construction. First, the stakeholders of a construction project, such as the owner, contractor, and supervisor, must reach an agreement regarding quality acceptance; this agreement includes the inspection object, inspection task, and quality constraints, which are ultimately reflected in the contractor's quality assurance system. Second, all agreements are transcoded from natural language to computerese; transcoding is a challenging task that is hardly handled by most frontline quality inspectors. The lack of standardization in modeling smart contracts also undermines the contracts' anti-risk capability. After translation, the agreements for quality acceptance are coded as computer-readable logical statements, such as IF–THEN judgment statements. The functions of smart contracts are activated after being deployed to the construction quality blockchain. These contracts can continuously monitor fluctuations in quality information in the chain. When the trigger conditions are initiated, the predefined responsive actions of smart contracts are automatically executed. Then, the intermediate information is documented on the chain; this process can be fulfilled without external disturbance.

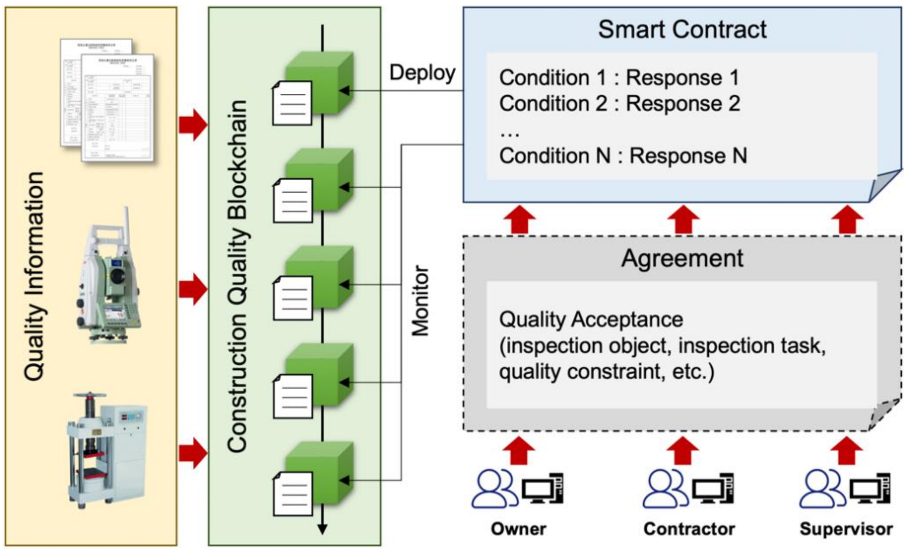


Figure 2. Conceptual scenario of smart contract-based quality acceptance in construction

3. Formal model of smart contracts for quality acceptance in construction

Formalization can avoid faults and ease the burden of inexperienced developers. Identifying and defining the fundamental constituents of the agreement for quality acceptance in construction are prerequisites of this process. Based on commercial contracts, Wang et al. [12] suggested that commitment, which they defined as a quintuple comprising a promisor, a promise, a premise, results, and time constraints, is the fundamental constituent for modeling smart contracts. However, this definition is broad and lacks pertinence for the scenario of quality acceptance.

3.1. Definition of the inspection task

In this study, CQA = {IT_0, IT_1, ⋯, IT_n} is defined as the agreement for quality acceptance that must be implemented in a construction project by the project stakeholders. IT refers to the inspection task, which is the fundamental constituent of CQA, and it can be formally defined as a sextuple.

$$IT_i = \left(R_i, A_i, CO_i, II_i, IR_i, AR_i\right) \in CQA,\ i = 0,1,\cdots,n$$

where

- R_i is the responsible person for IT_i, and this person must supervise self-inspection and issue inspection results;
- A_i is the accepter of IT_i and needs to recheck the inspection results issued by R_i and provide the acceptance results;
- CO_i is the checking object of IT_i, which can be a batch of building resources/products or a construction activity;
- II_i = {$II_{i,0}$, $II_{i,1}$, ⋯, $II_{i,m}$} is a collection of inspection items, where $II_{i,j}$ (j = 0, 1, ⋯, m) is a sole inspection item that is defined in detail hereinafter;
- IR_i is the inspection result of IT_i issued by R_i, who is responsible for the integrality and authenticity of IR_i;
- AR_i is the acceptance result of IT_i issued by A_i when CO_i can be accepted.

The sole inspection item $II_{i,j}$ is defined as

$$II_{i,j} = \left(R_{i,j}, QA_{i,j}, QC_{i,j}, QI_{i,j}, IR_{i,j}\right) \in II_i,\ j = 0,1,\cdots,m$$

where

- $R_{i,j}$ is the responsible person for $II_{i,j}$ who needs to capture and truthfully report on-site data and provide the inspection result of the inspection item;
- $QA_{i,j}$ is one of the quality attributes of CO_i that must be inspected in $II_{i,j}$;
- $QC_{i,j}$ is the quality constraint that $QA_{i,j}$ must satisfy to achieve the holistic quality goal of the project; related regulations, construction plans, and other items are the main sources and references for setting this constraint;
- $QI_{i,j}$ is the raw quality information captured for $II_{i,j}$ that should be documented in a traceable and tamper-proof manner for project management;
- $IR_{i,j}$ is the inspection result of $II_{i,j}$ issued by $R_{i,j}$.

3.2. Formal definition of smart contracts

CQA must be further encoded as a language-specific (e.g., Go- or Solidity-based) smart contract before deployment to the construction quality blockchain. The programming and deployment processes vary among different underlying blockchain infrastructure. Nonetheless, smart contracts are executable programs that switch between states on the basis of the data input, which can be described using FSM, a discrete mathematical method for modeling the state transition of a system [8]. In this study, FSM is used to formalize the smart contracts.

A smart contract for quality acceptance in construction (SC_CQA) is an FSM, which can be formally defined as

$$SC\_CQA = \left(S, s_0, F, \mathrm{S}, d\right)$$

where

- $S = \{s_0, s_1, \cdots, s_n\}$ is the state set of SC_CQA involved in its smart contract life cycle;
- s_0 is the initial state of SC_CQA ($s_0 \in S$);
- F is the set of terminal states of SC_CQA ($F \subset S$);
- Σ is the set of input events of quality information;
- δ is the set of state transition functions ($\delta: S \times \Sigma \rightarrow S$).

Figure 3 presents the state transition diagram of SC_CQA. The diagram can describe the various states and transitions between the states of SC_CQA during its life cycle. SC_CQA has a composite state, which is composed of multiple concurrent substates (as shown in Figure 3) because several inspection tasks of a CQA are performed concurrently without interfering each other. Each concurrent substate of SC_CQA is used to describe the state of an inspection task. These concurrent substates are activated synchronously under certain conditions; if and only if they are all completed, then the SC_CQA enters the next state. Similarly, an inspection task also has a composite state because an inspection task has multiple concurrent inspection items, and each inspection item has multiple states. The state set and the input event set of SC_CQA are defined in Tables 1 and 2, respectively.

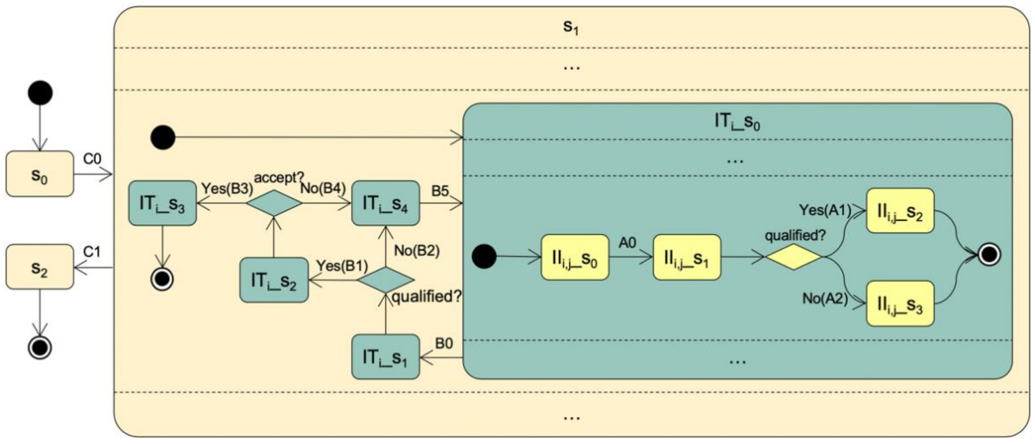


Figure 3. State transition diagram of the SC_CQA

Table 1. State set of SC_CQA

State	Description
II_s_0	Uncompleted state where the on-site checking information of $II_{i,j}$ is being collected and documented
II_s_1	Completed state where the requisite checking information of $II_{i,j}$ has been fully documented on the construction quality blockchain
II_s_2	Qualified state where $QA_{i,j}$ can satisfy $QC_{i,j}$ and therefore receives a “qualified” inspection result from $R_{i,j}$
II_s_3	Unqualified state where $QA_{i,j}$ cannot satisfy $QC_{i,j}$ and therefore receives an “unqualified” inspection result from $R_{i,j}$
IT_s_0	Uncompleted state where at least one more inspection item of IT_i has yet to be completed
IT_s_1	Completed state where all inspection items of IT_i are completed and waiting for the overall inspection result to be issued by R_i
IT_s_2	Qualified state where R_i issues a “qualified” result for CO_i and IT_i is waiting for the acceptance result from A_i
IT_s_3	Accepted state where A_i issues an “accepted” result for CO_i
IT_s_4	Rectified state where CO_i is identified as unqualified by R_i or A_i and must be rectified and checked again
s_0	Initial state
s_1	Running state where at least one inspection task of CQA has yet to be completed
s_2	Completed state where all inspection tasks of CQA are completed and accepted

Table 2. Input event set Σ

Event	Description
A0	All requisite checking information has been submitted.
A1	$II_{i,j}$ passes the quality compliance inspection.
A2	$II_{i,j}$ does not pass the quality compliance inspection.
B0	All inspection items have been completed.
B1	IT_i passes the overall quality inspection.
B2	IT_i does not pass the overall quality inspection and must be rectified.
B3	IT_i passes the acceptance.
B4	IT_i does not pass the acceptance and must be rectified.
B5	The rectification of IT_i is completed and reinspection begins.
C0	SC_CQA starts to run.
C1	All inspection tasks of SC_CQA are completed.

4. Case study

Support for smart contracts has become the fundamental function of blockchain infrastructure. This study takes the Hyperledger Fabric blockchain framework as a case study to demonstrate the implementation of SC_CQA. Hyperledger is an open-source project launched by the Linux Foundation for blockchain-based industrial applications, and Fabric is one of the permissioned blockchain frameworks in the Hyperledger project.

On the basis of the Hyperledger Fabric framework, a consortium blockchain can be established among the stakeholders of construction projects in a specific region to manage quality information. Using the real-name registration provided by the built-in membership service provider module rather than maintaining anonymity allows the information in the construction quality blockchain to be traced to specific individuals. The certificate authority module allows the establishment of a permission hierarchy in a project, such that each person's permission can be predefined by his/her superior. Many modularized consensus algorithms, such as Kafka and practical Byzantine fault tolerance, can ensure the consistency of quality information among peers in the construction quality blockchain.

Furthermore, chaincodes (i.e., smart contracts in the Hyperledger-based blockchain) enable the blockchain to perform the complex business logic of quality acceptance (Figure 4). Project stakeholders can deploy a chaincode for quality acceptance through the user interface of the construction quality blockchain. Once deployed, the chaincode acquires a unique address and exclusively runs on the blockchain network. In the subsequent management process, the project stakeholders cannot directly process the quality information in the blockchain, and the chaincode serves as a bridge. Through the user interface, a project stakeholder can send an input event to the deployed chaincode in accordance with the needs of the management. Then, the chaincode switches to a new state on the basis of the current state, and the predefined state transition rules and sends the relevant quality information to the latest block for documentation. For example, during the construction of a cast-in-situ bored pile inspection lot, the quality inspector of the contractor must measure the hole depth and input the data through the user interface. Subsequently, the state of the hole depth inspection item, which is in the chaincode of the cast-in-situ bored pile inspection lot, switches from uncompleted to completed, and the hole depth is sent by the chaincode to the latest block. Figure 5 presents a code snippet of a chaincodes for quality acceptance based on the Go language, which defines the input events of SC_CQA, as well as the starting and terminal states of these events.

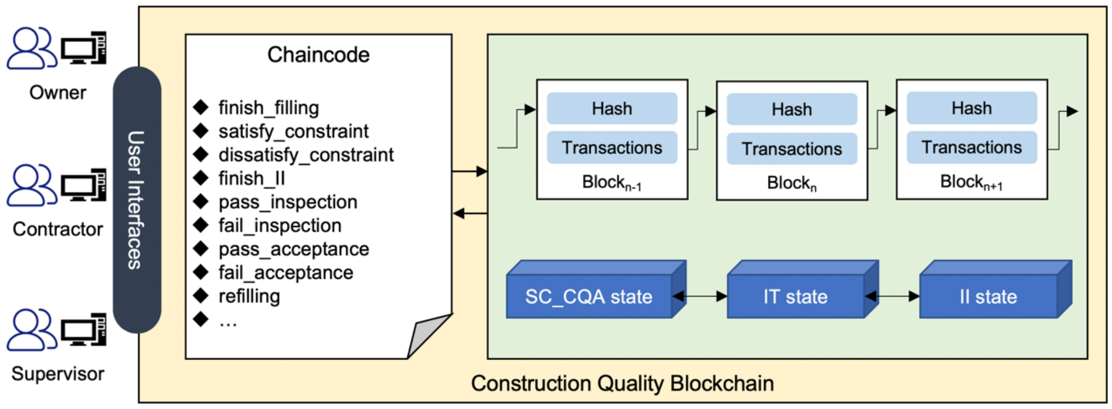
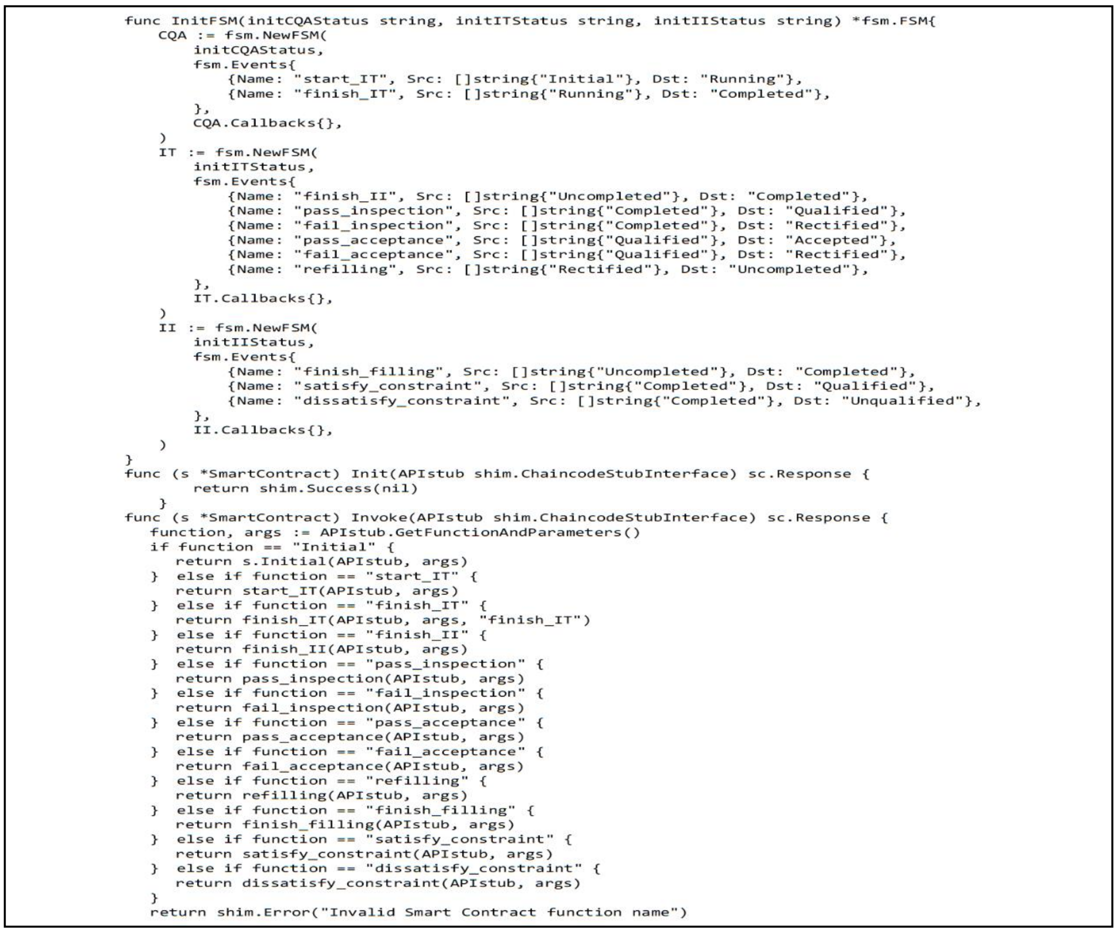


Figure 4. Smart contract for quality acceptance in construction



```
func InitFSM(initCQAStatus string, initITStatus string, initIIStatus string) *fsm.FSM{
    CQA := fsm.NewFSM(
        initCQAStatus,
        fsm.Events{
            {Name: "start_IT", Src: []string{"Initial"}, Dst: "Running"},
            {Name: "finish_IT", Src: []string{"Running"}, Dst: "Completed"},
        },
        CQA.Callbacks{},
    )
    IT := fsm.NewFSM(
        initITStatus,
        fsm.Events{
            {Name: "finish_II", Src: []string{"Uncompleted"}, Dst: "Completed"},
            {Name: "pass_inspection", Src: []string{"Completed"}, Dst: "Qualified"},
            {Name: "fail_inspection", Src: []string{"Completed"}, Dst: "Rectified"},
            {Name: "pass_acceptance", Src: []string{"Qualified"}, Dst: "Accepted"},
            {Name: "fail_acceptance", Src: []string{"Qualified"}, Dst: "Rectified"},
            {Name: "refilling", Src: []string{"Rectified"}, Dst: "Uncompleted"},
        },
        IT.Callbacks{},
    )
    II := fsm.NewFSM(
        initIIStatus,
        fsm.Events{
            {Name: "finish_filling", Src: []string{"Uncompleted"}, Dst: "Completed"},
            {Name: "satisfy_constraint", Src: []string{"Completed"}, Dst: "Qualified"},
            {Name: "dissatisfy_constraint", Src: []string{"Completed"}, Dst: "Unqualified"},
        },
        II.Callbacks{},
    )
}
func (s *SmartContract) Init(APIstub shim.ChaincodeStubInterface) sc.Response {
        return shim.Success(nil)
    }
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {
   function, args := APIstub.GetFunctionAndParameters()
   if function == "Initial" {
      return s.Initial(APIstub, args)
   }  else if function == "start_IT" {
      return start_IT(APIstub, args)
   }  else if function == "finish_IT" {
      return finish_IT(APIstub, args, "finish_IT")
   }  else if function == "finish_II" {
      return finish_II(APIstub, args)
   }  else if function == "pass_inspection" {
      return pass_inspection(APIstub, args)
   }  else if function == "fail_inspection" {
      return fail_inspection(APIstub, args)
   }  else if function == "pass_acceptance" {
      return pass_acceptance(APIstub, args)
   }  else if function == "fail_acceptance" {
      return fail_acceptance(APIstub, args)
   }  else if function == "refilling" {
      return refilling(APIstub, args)
   }  else if function == "finish_filling" {
      return finish_filling(APIstub, args)
   }  else if function == "satisfy_constraint" {
      return satisfy_constraint(APIstub, args)
   }  else if function == "dissatisfy_constraint" {
      return dissatisfy_constraint(APIstub, args)
   }
   return shim.Error("Invalid Smart Contract function name")
```

Figure 5. Code snippet of the smart contract for quality acceptance

5. Conclusion

Blockchains and smart contracts are promising tools for improving quality management in construction. The former provides quality information and tamper resistance, and the latter can (semi-)automate the business logic of quality management. However, the coding and deployment of smart contracts for quality management are high-threshold tasks for quality managers. To address this issue, this study proposes an FSM-based formal model of smart contracts for quality acceptance in construction. A Hyperledger-based case study is performed to demonstrate the functions of the proposed model. However, the effectiveness of the proposed model requires further evaluation using practical evidence, which is a limitation of this research.

Smart contract technology is still a novelty in the construction industry, and our work provides insights into the industrial application of smart contracts. Our future work will include the method of generating smart contract codes for quality management by mapping the FSM model and smart contracts. Such an investigation would further lower the application threshold of smart contracts for quality management personnel.

6. References

[1] M. Hamilton, Blockchain distributed ledger technology: An introduction and focus on smart contracts, Journal of Corporate Accounting and Finance 31 (2019) 7-12. https://doi.org/10.1002/jcaf.22421
[2] S. Perera, S. Nanayakkara, M.N.N. Rodrigo, S. Senaratne, R. Weinand, Blockchain technology: Is it hype or real in the construction industry?, Journal of Industrial Information Integration 17 (2020) 100125. https://doi.org/10.1016/j.jii.2020.100125
[3] J. Li, D. Greenwood, M. Kassem, Blockchain in the built environment and construction industry: A systematic review, conceptual models and practical use cases, Automation in Construction 102 (2019) 288-307. https://doi.org/10.1016/j.autcon.2019.02.005
[4] Z. Wang, T. Wang, H. Hu, J. Gong, X. Ren, Q. Xiao, Blockchain-based framework for improving supply chain traceability and information sharing in precast construction, Automation in Construction 111 (2020) 103063. https://doi.org/10.1016/j.autcon.2019.103063
[5] F. Elghaish, S. Abrishami, M. R. Hosseini, Integrated project delivery with blockchain: An automated financial system, Automation in Construction 114 (2020) 103182. https://doi.org/10.1016/j.autcon.2020.103182

[6] H. Chong, A. Diamantopoulos, Integrating advanced technologies to uphold security of payment: Data flow diagram, Automation in Construction 114 (2020) 103158. https://doi.org/10.1016/j.autcon.2020.103158

[7] A. Singh, R.M. Parizi, Q. Zhang, K.K.R. Choo, A. Dehghantanha, Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities, Computers & Security 88 (2020) 101654. https://doi.org/10.1016/j.cose.2019.101654

[8] X.M. Bai, Z.J. Cheng, Z.B. Duan, K. Hu, Formal Modeling and Verification of Smart Contracts, Proceedings of 2018 7th International Conference on Software and Computer Applications (Icsca 2018) (2018) 322-326. https://doi.org/10.1145/3185089.3185138

[9] A. Mavridou, A. Laszka, Tool Demonstration: FSolidM for Designing Secure Ethereum Smart Contracts, Principles of Security and Trust (Post 2018) 10804 (2018) 270-277. https://doi.org/10.1007/978-3-319-89722-6_11

[10] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts, 2016 IEEE Symposium on Security and Privacy (SP), (2016) 839-858. https://doi.org/10.1109/SP.2016.55

[11] N. Szabo, Smart Contracts (1994). http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, Accessed May 28, 2020

[12] P. Wang, H. Yang, J. Meng, J. Chen, X. Du, Formal Definition for Classical Smart Contracts and Reference Implementation, Journal of Software 30 (9) (2019) 2608-2619. https://doi.org/10.13328/j.cnki.jos.005773